Search Results

Search Results for: **[dynamic AND compile AND annotation]**
Found **1,539** of **105,778 searched.** → Rerun within the Portal
Warning: Maximum result set of 200 exceeded. Consider refining.

Search within Results

> Advanced Search > Search Help/Tips

Sort by: Title Publication Publication Date Score

Results 1 - 20 of 200 short listing

Prev Page 1 2 3 4 5 6 7 8 9 10 Next Page

1 Calpa: a tool for automating selective dynamic compilation 100%
Markus Mock , Craig Chambers , Susan J. Eggers
Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture
December 2000

2 Annotation-directed run-time specialization in C 100%
Brian Grant , Markus Mock , Matthai Philipose , Craig Chambers , Susan J. Eggers
ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation December 1997
Volume 32 Issue 12

3 The benefits and costs of DyC's run-time optimizations 100%
Brian Grant , Markus Mock , Matthai Philipose , Craig Chambers , Susan J. Eggers
ACM Transactions on Programming Languages and Systems (TOPLAS) September 2000
Volume 22 Issue 5
DyC selectively dynamically compiles programs during their execution, utilizing the run-time-computed values of variables and data structures to apply optimizations that are based on partial evaluation. The dynamic optimizations are preplanned at static compile time in order to reduce their run-time cost; we call this staging. DyC's staged optimizations include (1) an advanced binding-time analysis that supports polyvariant specialization (enabling both single-way and multi ...

4 Using annotations to reduce dynamic optimization time 100%
Chandra Krintz , Brad Calder
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN'01 conference on Programming language design and implementation May 2001
Volume 36 Issue 5

Dynamic compilation and optimization are widely used in heterogenous computing environments, in which an intermediate form of the code is compiled to native code during execution. An important trade off exists between the amount of time spent dynamically optimizing the program and the running time of the program. The time to perform dynamic optimizations can cause significant delays during execution and also prohibit performance gains that result from more complex optimization.

5 An evaluation of staged run-time optimizations in DyC 100%
Brian Grant , Matthai Philipose , Markus Mock , Craig Chambers , Susan J. Eggers
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN '99 conference on Programming language design and implementation May 1999
Volume 34 Issue 5

6 Fast, effective dynamic compilation 99%
Joel Auslander , Matthai Philipose , Craig Chambers , Susan J. Eggers , Brian N. Bershad
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN '96 conference on Programming language design and implementation May 1996
Volume 31 Issue 5

7 Partial evaluation and separate compilation 98%
Rogardt Heldal , John Hughes
ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation December 1997
Volume 32 Issue 12

8 Parallel execution of prolog programs: a survey 97%
Gopal Gupta , Enrico Pontelli , Khayri A.M. Ali , Mats Carlsson , Manuel V. Hermenegildo
ACM Transactions on Programming Languages and Systems (TOPLAS) July 2001
Volume 23 Issue 4

Since the early days of logic programming, researchers in the field realized the potential for exploitation of parallelism present in the execution of logic programs. Their high-level nature, the presence of nondeterminism, and their referential transparency, among other characteristics, make logic programs interesting candidates for obtaining speedups through parallel execution. At the same time, the fact that the typical applications of logic programming frequently involve irregular computatio ...

9 Dynamic typing as staged type inference 97%
Mark Shields , Tim Sheard , Simon Peyton Jones
Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages January 1998

10 Java annotation-aware just-in-time (AJIT) complilation system 96%
Ana Azevedo , Alex Nicolau , Joe Hummel
Proceedings of the ACM 1999 conference on Java Grande June 1999

11 Supporting dynamic data structures on distributed-memory machines 95%
Anne Rogers , Martin C. Carlisle , John H. Reppy , Laurie J. Hendren
ACM Transactions on Programming Languages and Systems (TOPLAS) March 1995
Volume 17 Issue 2

Compiling for distributed-memory machines has been a very active research area in recent years. Much of this work has concentrated on programs that use arrays as their primary data structures. To date, little work has been done to address the problem of supporting programs that use pointer-based dynamic data structures. The techniques developed for supporting SPMD execution of array-based programs rely on the fact that arrays are statically defined and directly addressable. Recursive data s ...

12 Separation constraint partitioning: a new algorithm for partitioning non-strict programs into sequential threads 95%
Klaus E. Schauser , David E. Culler , Seth C. Goldstein
Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages January 1995

In this paper we present substantially improved thread partitioning algorithms for modern implicitly parallel languages. We present a new block partitioning algorithm, separation constraint partitioning, which is both more powerful and more flexible than previous algorithms. Our algorithm is guaranteed to derive maximal threads. We present a theoretical framework for proving the correctness of our partitioning approach, and we show how separation constraint partitioning mak ...

13 Region-based memory management in cyclone 95%
Dan Grossman , Greg Morrisett , Trevor Jim , Michael Hicks , Yanling Wang , James Cheney
ACM SIGPLAN Notices , Proceeding of the ACM SIGPLAN 2002 Conference on Programming language design and implementation May 2002
Volume 37 Issue 5

Cyclone is a type-safe programming language derived from C. The primary design goal of Cyclone is to let programmers control data representation and memory management without sacrificing type-safety. In this paper, we focus on the region-based memory management of Cyclone and its static typing discipline. The design incorporates several advancements, including support for region subtyping and a coherent integration with stack allocation and a garbage collector. To support separate compilation, C ...

14 Incremental partial evaluation: the key to high performance, modularity and portability in operating systems 95%
Charles Consel , Calton Pu , Jonathan Walpole
Proceedings of the ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation August 1993

15 Staged compilation 95%
Craig Chambers
ACM SIGPLAN Notices , Proceedings of the 2002 ACM SIGPLAN workshop on Partial evaluation and semantics-based program manipulation January 2002
Volume 37 Issue 3

Traditional compilers compile and optimize files separately, making worst-case assumptions about the program context in which a file is to be linked. More aggressive compilation architectures perform cross-file interprocedural or whole-program analyses, potentially producing much faster programs but substantially increasing the cost of compilation. Even more radical are systems that perform all compilation and optimization at run-time: such systems can optimize programs based on run-time program ...

16 Formalizing the safety of Java, the Java virtual machine, and Java card 94%
Pieter H. Hartel , Luc Moreau
ACM Computing Surveys (CSUR) December 2001
Volume 33 Issue 4

We review the existing literature on Java safety, emphasizing formal approaches, and the impact of Java safety on small footprint devices such as smartcards. The conclusion is that although a lot of good work has been done, a more concerted effort is needed to build a coherent set of machine-readable formal models of the whole of Java and its implementation. This is a formidable task but we believe it is essential to build trust in Java safety, and thence to achieve ITSEC level 6 or Common Crite ...

17 Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science 94%

William F. Atchison , Samuel D. Conte , John W. Hamblen , Thomas E. Hull , Thomas A. Keenan , William B. Kehl , Edward J. McCluskey , Silvio O. Navarro , Werner C. Rheinboldt , Earl J. Schweppe , William Viavant , David M. Young
Communications of the ACM March 1968
Volume 11 Issue 3

18 Guidance for the use of the Ada programming language in high integrity systems 94%
B. A. Wichmann
ACM SIGAda Ada Letters July 1998
Volume XVIII Issue 4

This paper is the current result of a study by the ISO HRG Rapporteur group which is being circulated for comment. Many people have contributed to this, but those who have either attended two recent meetings of group or have made substantial e-mail comments are: Praful V Bhansali (Boeing, USA), Alan Burns (University of York, UK), Bernard Carre' (Praxis Critical Systems, UK), Dan Craigen (ORA, Canada), Nick Johnson MoD, UK), Stephen Michell (Canada), Gilles Motet (DGEI/INSA, France), George Roma ...

19 Module-sensitive program specialisation 93%
Dirk Dussart , Rogardt Heldal , John Hughes
ACM SIGPLAN Notices , Proceedings of the 1997 ACM SIGPLAN conference on Programming language design and implementation May 1997
Volume 32 Issue 5

20 Mix ten years later 93%
Neil D. Jones
Proceedings of the ACM SIGPLAN Symposium on Partial evaluation and semantics-based program manipulation June 1995

Results 1 - 20 of 200 **short listing**

Prev Page 1 2 3 4 5 6 7 8 9 10 Next Page